

日 本 国 特 許 庁  
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日            2 0 0 3 年   7 月 1 4 日  
Date of Application:

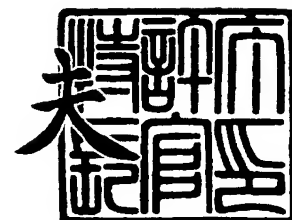
出 願 番 号            特 願 2 0 0 3 - 1 9 6 2 2 8  
Application Number:  
[ST. 10/C]:            [ J P 2 0 0 3 - 1 9 6 2 2 8 ]

出      願      人            株 式 会 社 リ コ ー  
Applicant(s):

2 0 0 3 年   7 月 2 8 日

特許庁長官  
Commissioner,  
Japan Patent Office

今 井 康



【書類名】 特許願

【整理番号】 0305119

【提出日】 平成15年 7月14日

【あて先】 特許庁長官 今井 康夫 殿

【国際特許分類】 G03G 21/00 370

【発明の名称】 画像形成装置及びラッピング処理方法並びにプログラム

【請求項の数】 34

【発明者】

【住所又は居所】 東京都大田区中馬込 1 丁目 3 番 6 号 株式会社リコー内

【氏名】 大石 勉

【発明者】

【住所又は居所】 東京都大田区中馬込 1 丁目 3 番 6 号 株式会社リコー内

【氏名】 中川 克彦

【発明者】

【住所又は居所】 東京都大田区中馬込 1 丁目 3 番 6 号 株式会社リコー内

【氏名】 秋吉 邦洋

【発明者】

【住所又は居所】 東京都大田区中馬込 1 丁目 3 番 6 号 株式会社リコー内

【氏名】 田中 浩行

【特許出願人】

【識別番号】 000006747

【氏名又は名称】 株式会社リコー

【代理人】

【識別番号】 100070150

【弁理士】

【氏名又は名称】 伊東 忠彦

【先の出願に基づく優先権主張】

【出願番号】 特願2002-211685

【出願日】 平成14年 7月19日

【手数料の表示】

【予納台帳番号】 002989

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9911477

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 画像形成装置及びラッピング処理方法並びにプログラム

【特許請求の範囲】

【請求項 1】 画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置において、

アプリケーションから呼び出される関数を変換し、変換後の関数を用いてコントロールサービスへの関数呼び出しを行うラッピング処理手段を備えたことを特徴とする画像形成装置。

【請求項 2】 前記ラッピング処理手段は、前記アプリケーションが前記コントロールサービスに対して使用する関数と、この関数に対応してコントロールサービスが有する関数との間でバージョン差がある場合に前記関数の変換を行う請求項 1 に記載の画像形成装置。

【請求項 3】 前記ラッピング処理手段は、前記コントロールサービスが有する関数のバージョンが変更されたことを示す情報を参照することにより、前記バージョン差があることを判断する請求項 2 に記載の画像形成装置。

【請求項 4】 前記ラッピング処理手段は、前記アプリケーションが前記コントロールサービスに対して使用する関数と、当該関数に対応するコントロールサービス側の関数との間で関数の数又は引数の数が異なる場合に、ダミーの関数又はダミーの引数を補填する変換を行う請求項 1 に記載の画像形成装置。

【請求項 5】 画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置において、

前記コントロールサービスが送信するメッセージの中から前記アプリケーションに通知するメッセージを取捨選択するラッピング処理手段を備えたことを特徴とする画像形成装置。

【請求項 6】 前記ラッピング処理手段は、前記アプリケーションに対して通知しない特定のメッセージを記録した情報を参照することにより、前記アプリケーションに対するメッセージを取捨選択する請求項 5 に記載の画像形成装置。

【請求項 7】 前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスを更に備え、

前記ラッピング処理手段は、前記仮想アプリケーションサービスに含まれる請求項 1 ないし 6 のうちいずれか 1 項に記載の画像形成装置。

【請求項 8】 前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを判断するバージョン管理手段を更に有する請求項 7 に記載の画像形成装置。

【請求項 9】 前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記バージョンが前記所定の範囲内であるか否かを判断する請求項 8 に記載の画像形成装置。

【請求項 10】 前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを、関数単位に判断するバージョン管理手段を更に有する請求項 7 に記載の画像形成装置。

【請求項 11】 前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数毎のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記関数のバージョンが前記所定の範囲内であるか否かを関数単位に判断する請求項 10 に記載の画像形成装置。

【請求項 12】 画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置が実行する処理の方法であって、

アプリケーションから呼び出される関数を変換し、変換後の関数を用いてコントロールサービスへの関数呼び出しを行うラッピング処理ステップを有することを特徴とする方法。

【請求項 13】 前記ラッピング処理ステップにおいて、前記アプリケーションが前記コントロールサービスに対して使用する関数と、この関数に対応してコントロールサービスが有する関数との間でバージョン差がある場合に関数の変換を行う請求項 12 に記載の方法。

【請求項 14】 前記ラッピング処理ステップにおいて、前記コントロールサービスが有する関数のバージョンが変更されたことを示す情報を参照することにより、前記バージョン差があることを判断する請求項 13 に記載の方法。

【請求項 15】 前記ラッピング処理ステップにおいて、前記アプリケーションが前記コントロールサービスに対して使用する関数と、当該関数に対応するコントロールサービス側の関数との間で関数の数又は引数の数が異なる場合に、ダミーの関数又はダミーの引数を補填する変換を行う請求項 12 に記載の方法。

【請求項 16】 画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置が実行する処理の方法であって、

前記コントロールサービスが送信するメッセージの中から前記アプリケーションに通知するメッセージを取捨選択するラッピング処理ステップを有することを特徴とする方法。

【請求項 17】 前記ラッピング処理ステップにおいて、前記アプリケーションに対して通知しない特定のメッセージを記録した情報を参照することにより、前記アプリケーションに対するメッセージを取捨選択する請求項 16 に記載の方法。

【請求項 18】 前記画像形成装置は、前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスを更に備え、前記ラッピング処理ステップは、前記仮想アプリケーションサービスが実行する請求項 12 ないし 17 のうちいずれか 1 項に記載の方法。

【請求項 19】 前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを判断するバージョン管理ステッ

プを更に有する請求項 1 8 に記載の方法。

【請求項 2 0】 前記バージョン管理ステップにおいて、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記バージョンが前記所定の範囲内であるか否かを判断する請求項 1 9 に記載の方法。

【請求項 2 1】 前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを、関数単位に判断するバージョン管理ステップを更に有する請求項 1 8 に記載の方法。

【請求項 2 2】 前記バージョン管理ステップにおいて、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数毎のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記関数のバージョンが前記所定の範囲内であるか否かを関数単位に判断する請求項 2 1 に記載の方法。

【請求項 2 3】 画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置を、

アプリケーションから呼び出される関数を変換し、変換後の関数を用いてコントロールサービスへの関数呼び出しを行うラッピング処理手段として機能させるプログラム。

【請求項 2 4】 前記ラッピング処理手段は、前記アプリケーションが前記コントロールサービスに対して使用する関数と、この関数に対応してコントロールサービスが有する関数との間でバージョン差がある場合に関数の変換を行う請求項 2 3 に記載のプログラム。

【請求項 2 5】 前記ラッピング処理手段は、前記コントロールサービスが有する関数のバージョンが変更されたことを示す情報を参照することにより、前記バージョン差があることを判断する請求項 2 4 に記載のプログラム。

【請求項 2 6】 前記ラッピング処理手段は、前記アプリケーションが前記

コントロールサービスに対して使用する関数と、当該関数に対応するコントロールサービス側の関数との間で関数の数又は引数の数が異なる場合に、ダミーの関数又はダミーの引数を補填する変換を行う請求項 2 3 に記載のプログラム。

【請求項 2 7】 画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置を、

前記コントロールサービスが送信するメッセージの中から前記アプリケーションに通知するメッセージを取捨選択するラッピング処理手段として機能させるプログラム。

【請求項 2 8】 前記ラッピング処理手段は、前記アプリケーションに対して通知しない特定のメッセージを記録した情報を参照することにより、前記アプリケーションに対するメッセージを取捨選択する請求項 2 7 に記載のプログラム。

【請求項 2 9】 前記プログラムは、前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスのプログラムに含まれる請求項 2 3 ないし 2 8 のうちいずれか 1 項に記載のプログラム。

【請求項 3 0】 前記画像形成装置を、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを判断するバージョン管理手段として更に機能させる請求項 2 9 に記載のプログラム。

【請求項 3 1】 前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記バージョンが前記所定の範囲内であるか否かを判断する請求項 3 0 に記載のプログラム。

【請求項 3 2】 前記画像形成装置を、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数のバージョンが、前記仮想アプリ



ケーションサービスがサポートできる所定の範囲内であるか否かを、関数単位に判断するバージョン管理手段として更に機能させる請求項 2 9 に記載のプログラム。

【請求項 3 3】 前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数毎のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記関数のバージョンが前記所定の範囲内であるか否かを関数単位に判断する請求項 3 2 に記載のプログラム。

【請求項 3 4】 請求項 2 3 ないし 3 3 のうちいずれか 1 項に記載のプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

この発明は、コピー、プリンタ、スキャナおよびファクシミリなどの画像形成処理にかかるユーザサービスを提供する画像形成装置に関し、特に、アプリケーションとアプリケーションが利用するサービスとの間でバージョン差が生じた場合に、バージョン差を吸収できる画像形成装置に関するものである。

【0 0 0 2】

【従来の技術】

近年では、プリンタ、コピー、ファクシミリ、スキャナなどの各装置の機能を 1 つの筐体内に収納した画像形成装置（以下、「複合機」という。）が一般的に知られている。この複合機は、1 つの筐体内に表示部、印刷部および撮像部などを設けるとともに、プリンタ、コピーおよびファクシミリ装置にそれぞれ対応する 3 種類のソフトウェアを設け、これらのソフトウェアを切り替えることによって、当該装置をプリンタ、コピー、スキャナ又はファクシミリ装置として動作させるものである。

【0 0 0 3】

このような従来の複合機では、プリンタ、コピー、ファクシミリ、スキャナなどの各機能単位でアプリケーションプログラムが起動され、ハードウェア資源に

アクセスする機能を持ち合わせている。その際、アプリケーションプログラムが前提としているオペレーティングシステム（OS）のバージョンと、実際のオペレーティングシステム（OS）のバージョンとが同じことが前提となるが、例えば、OSをバージョンアップしてアプリケーションとの間でバージョン差が生じた場合、今まで使えていた機能が使えなくなったり、アプリケーションそのものが起動しなくなったりすることがある。

#### 【0004】

このため、従来の複合機では、OSなどをバージョンアップすると、それに伴ってアプリケーションの方もコンパイルし直すことで、常に対応したバージョン関係にあることが要請されている。

#### 【0005】

##### 【特許文献1】

特開2002-82806号公報

#### 【0006】

##### 【発明が解決しようとする課題】

ところで、このような従来の複合機では、プリンタ、コピー、スキャナおよびファクシミリ装置に対応するソフトウェアがそれぞれ別個に設けられているため、各ソフトウェアの開発に多大の時間を要する。このため、出願人は、表示部、印刷部および撮像部などの画像形成処理で使用されるハードウェア資源を有し、プリンタ、コピー又はファクシミリなどの各ユーザサービスにそれぞれ固有の処理を行うアプリケーションを複数搭載し、これらのアプリケーションとハードウェア資源との間に介在して、ユーザサービスを提供する際に、アプリケーションの少なくとも2つが共通的に必要とするハードウェア資源の管理、実行制御並びに画像形成処理を行う各種コントロールサービスからなるプラットフォームを備えた画像形成装置（複合機）を発明した。

#### 【0007】

このような新規な複合機では、アプリケーションとコントロールサービスとを別個に設けているため、複合機の出荷後にユーザもしくは第三者であるサードベンダが新規なアプリケーションを開発して複合機に搭載することが可能となり、

これによって多種多様な機能を提供することが可能となる。

#### 【0008】

このような新規な複合機では、アプリケーションの少なくとも2つが共通に必要とするサービスを提供するコントロールサービスをアプリケーションと別個に設けた構成となっているため、新規アプリケーションを開発する場合には、各種コントロールサービスとのプロセス間通信を実現する処理をソースコードで記述する必要がある。しかしながら、新規アプリケーションを開発する場合、各コントロールサービスが提供する関数やメッセージなどを正確に把握した上で、予め規定された手順で記述しなければならず、各コントロールサービスやアプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェース（API）をデバッグや機能追加によって繰り返しバージョンアップすると、ベンダーはどのバージョンに合わせてアプリケーション開発すればよいかが非常に分かり難くなり、アプリケーションの開発が難しくなる。また、バージョン差のあるアプリケーションを実行させることにより、システムに影響を及ぼす恐れもある。このことは、固定的な複数の機能を寄せ集めた従来の複合機では問題にならなかった新規な課題である。

#### 【0009】

また、各コントロールサービスとアプリケーションのインタフェース（API）のすべてを、新規アプリケーションを開発するサードベンダなどの第三者に開示することは、プログラムの秘匿性、システムの安全性の観点から好ましくない。

#### 【0010】

この発明は上記に鑑みてなされたもので、バージョンアップ等によりアプリケーションが使用している関数とコントロールサービス側の関数とで差異が生じた場合でも、アプリケーションのソースプログラムを書き換えることなく、関数呼び出しを行うことを可能とする技術を提供することを目的とする。また、コントロールサービスからアプリに通知されるメッセージを秘匿する技術を提供することを目的とする。

#### 【0011】

**【課題を解決するための手段】**

上記目的を達成するため、請求項 1 にかかる発明は、画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置において、アプリケーションから呼び出される関数を変換し、変換後の関数を用いてコントロールサービスへの関数呼び出しを行うラッピング処理手段を備える。

**【0012】**

本発明によれば、アプリケーションから呼び出された関数を変換して、関数呼び出しを行うので、アプリケーションが使用している関数とコントロールサービス側の関数とで差異が生じた場合でも、アプリケーションのソースプログラムを書き換えることなく、関数呼び出しを行うことが可能となる。

**【0013】**

請求項 2 に記載の発明は、請求項 1 の記載において、前記ラッピング処理手段は、前記アプリケーションが前記コントロールサービスに対して使用する関数と、コントロールサービスが有する関数との間でバージョン差がある場合に関数の変換を行う請求項 1 に記載の画像形成装置。

**【0014】**

本発明によれば、例えば、コントロールサービス側の関数のバージョンアップがあり、バージョン差が生じている場合に関数の変換を行う。これにより、バージョン不整合による不具合発生を防止できる。  
することができる。

**【0015】**

請求項 3 に記載の発明は、請求項 2 の記載において、前記ラッピング処理手段は、前記コントロールサービスが有する関数のバージョンが変更されたことを示す情報を参照することにより、前記バージョン差があることを判断する。これによりバージョン差があること正確に判断できる。

**【0016】**

請求項 4 に記載の発明は、請求項 1 の記載において、前記ラッピング処理手段は、前記アプリケーションが前記コントロールサービスに対して使用する関数と

、当該関数に対応するコントロールサービス側の関数との間で関数の数又は引数の数が異なる場合に、ダミーの関数又はダミーの引数を補填する変換を行う。

【0017】

アプリケーションが前記コントロールサービスに対して使用する関数と、当該関数に対応するコントロールサービス側の関数との間で関数の数又は引数の数が異なる場合には、関数呼び出しを行うことができなくなるが、本発明によれば、増加した関数や関数の引数に対応するダミーの関数又は関数の引数を補填することによって関数呼び出しが可能となる。

【0018】

請求項5に記載の発明は、画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置において、前記コントロールサービスが送信するメッセージの中から前記アプリケーションに通知するメッセージを取捨選択するラッピング処理手段を備えたものである。

【0019】

本発明によれば、第三者に対してはコントロールサービスとの間のインタフェースを隠蔽することができる。

【0020】

請求項6に記載の発明は、請求項5の記載において、前記ラッピング処理手段は、前記アプリケーションに対して通知しない特定のメッセージを記録した情報を参照することにより、前記アプリケーションに対するメッセージを取捨選択する。

【0021】

本発明によれば、アプリケーションに対して通知しない特定のメッセージを記録した情報を参照するようにしたので、当該情報に所望のメッセージを記録しておくことにより、容易にメッセージを取捨選択を行うことが可能となる。

【0022】

請求項7に記載の発明は、請求項1ないし6のうちいずれか1項の記載において、前記コントロールサービスをサーバとしたクライアントプロセスとして動作

し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスを更に備え、前記ラッピング処理手段は、前記仮想アプリケーションサービスに含まれるというものである。本発明も、請求項 1 等と同様の効果を奏する。

#### 【0023】

請求項 8 に記載の発明は、請求項 7 の記載において、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを判断するバージョン管理手段を更に有するものである。

#### 【0024】

本発明によれば、アプリケーションが仮想アプリケーションサービスに対して使用する関数全体のバージョンが、仮想アプリケーションサービスのサポート範囲内か否かを判断できるため、例えば、アプリケーションを実際に使用する前にこの判断を行うことにより、アプリケーションが仮想アプリケーションサービス上で使用できるもの否かを判断でき、障害の発生を未然に防止することができる。

#### 【0025】

請求項 9 に記載の発明は、請求項 8 の記載において、前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記バージョンが前記所定の範囲内であるか否かを判断するものである。本発明によれば、容易に前記バージョンが前記所定の範囲内であるか否かを判断できる。

#### 【0026】

請求項 10 に記載の発明は、請求項 7 の記載において、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを、関数単位に判断するバージョン管理手段を更に有するものである。

#### 【0027】

本発明によれば、関数単位に判断できるので、例えば、アプリケーションが使用していない関数がバージョンアップされても、このことに影響を受けずに判断を行うことができる。また、サポートが不可能なバージョン差がある関数を容易に特定することができる。

#### 【0028】

請求項11に記載の発明は、請求項10に記載において、前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数毎のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記関数のバージョンが前記所定の範囲内であるか否かを関数単位に判断する。本発明によれば、容易に上記の判断を行うことができる。

#### 【0029】

請求項12～22に記載の発明は、上記の画像形成装置における処理の方法の発明である。請求項23～33に記載の発明は、上記の画像形成装置における処理の手順を画像形成装置に実行させるためのプログラムである。また、請求項34に記載の発明は、そのプログラムを格納したコンピュータ読み取り可能な記録媒体である。これらの発明によっても上記の画像形成装置の発明と同様の効果を奏する。

#### 【0030】

##### 【発明の実施の形態】

以下に添付図面を参照して、本発明の好適な実施の形態を詳細に説明する。

#### 【0031】

##### (実施の形態1)

図1は、この発明の実施の形態1である画像形成装置（以下、「複合機」という）の構成を示すブロック図である。図1に示すように、複合機100は、白黒レーザプリンタ（B&W LP）101と、カラーレーザプリンタ（Color LP）102と、スキャナ、ファクシミリ、ハードディスク、メモリ、ネットワークインタフェースなどのハードウェアリソース103を有するとともに、プラットフォーム120とアプリケーション130と仮想アプリケーションサービス（VAS：Vi

rtual Application Service) 140から構成されるソフトウェア群110を備えている。

#### 【0032】

仮想アプリケーションサービス (VAS) 140は、アプリケーション130とプラットフォーム120の間に配置される。VAS140は、アプリから見るとプラットフォーム120のサービス層として認識され、サービス層から見るとアプリとして認識されるように登録される。また、VAS140は、アプリに対してはサーバプロセスとして動作し、各コントロールサービスに対してはクライアントプロセスとして動作する。

#### 【0033】

このVAS140の第1の基本機能は、アプリケーションがコントロールサービスの機能を利用するために使用しているAPIとしての関数と、コントロールサービスが提供するAPIとしての関数との間でのバージョン差を吸収する機能である。この機能により、バージョン差があったとしても、アプリケーション130を再コンパイルすることなくそのまま関数呼び出しを行うことができる。また、コントロールサービスからのメッセージを取捨選択することによってプラットフォーム120を隠蔽する機能もある。これらをラッピング機能という。

#### 【0034】

VAS140の第2の基本機能は、アプリがVAS140に対して使用する関数のバージョンと、VAS140が有する関数のバージョンとのバージョン差を検出し、その検出されたバージョン差がVAS140によってサポート可能な範囲か否かを判定して、その判定結果をアプリに通知する機能を有している。これにより、例えばアプリ動作前にバージョンが適切かどうかを判断できる。これをバージョン管理機能という。

#### 【0035】

なお、VAS140のプログラムを、例えば、SD (Secure Digital) カードに格納して、そこからVAS140を起動することも可能である。また、ネットワーク接続されたサーバにVAS140のプログラムを格納しておき、そこからネットワーク経由でVAS140を複合機100にインストール、もしくは、起



動することが可能である。

#### 【0036】

プラットフォーム120は、アプリケーションからの処理要求を解釈してハードウェア資源の獲得要求を発生させるコントロールサービスと、一又は複数のハードウェア資源の管理を行い、コントロールサービスからの獲得要求を調停するシステムリソースマネージャ (SRM) 123と、汎用OS 121とを有する。

#### 【0037】

コントロールサービスは、複数のサービスモジュールから形成され、SCS (システムコントロールサービス) 122と、ECS (エンジンコントロールサービス) 124と、MCS (メモリコントロールサービス) 125と、OCS (オペレーションパネルコントロールサービス) 126と、FCS (ファックスコントロールサービス) 127と、NCS (ネットワークコントロールサービス) 128とから構成される。なお、このプラットフォーム120の各サービスは、予め定義された関数により前記アプリケーション130から処理要求を受信可能とするアプリケーションプログラムインタフェース (API) を有している。

#### 【0038】

汎用OS 121は、UNIX (登録商標) などの汎用オペレーティングシステムであり、プラットフォーム120並びにアプリケーション130の各ソフトウェアをそれぞれプロセスとして並列実行する。

#### 【0039】

SRM123のプロセスは、SCS122とともにシステムの制御およびリソースの管理を行うものである。SRM123のプロセスは、スキャナ部やプリンタ部などのエンジン、メモリ、HDDファイル、ホストI/O (セントロI/F、ネットワークI/F、IEEE1394 I/F、RS232C I/Fなど) のハードウェア資源を利用する上位層からの要求にしたがって調停を行い、実行制御する。

#### 【0040】

具体的には、このSRM123は、要求されたハードウェア資源が利用可能であるか (他の要求により利用されていないかどうか) を判断し、利用可能であれ

ば要求されたハードウェア資源が利用可能である旨を上位層に伝える。また、SRM123は、上位層からの要求に対してハードウェア資源の利用スケジューリングを行い、要求内容（例えば、プリンタエンジンにより紙搬送と作像動作、メモリ確保、ファイル生成など）を直接実施している。

#### 【0041】

SCS122のプロセスは、アプリ管理、操作部制御、システム画面表示、LED表示、リソース管理、割り込みアプリ制御などを行う。

#### 【0042】

ECS124のプロセスは、白黒レーザプリンタ（B&W LP）101、カラーレーザプリンタ（Color LP）102、スキャナ、ファクシミリなどからなるハードウェアリソース103のエンジンの制御を行う。

#### 【0043】

MCS125のプロセスは、画像メモリの取得および解放、ハードディスク装置（HDD）の利用、画像データの圧縮および伸張などを行う。

#### 【0044】

FCS127のプロセスは、システムコントローラの各アプリ層からPSTN／ISDN網を利用したファクシミリ送受信、BKM（バックアップSRAM）で管理されている各種ファクシミリデータの登録／引用、ファクシミリ読みとり、ファクシミリ受信印刷、融合送受信を行うためのAPIを提供する。

#### 【0045】

NCS128のプロセスは、ネットワークI/Oを必要とするアプリケーションに対して共通に利用できるサービスを提供するためのプロセスであり、ネットワーク側から各プロトコルによって受信したデータを各アプリケーションに振り分けたり、アプリケーションからデータをネットワーク側に送信する際の仲介を行う。具体的には、ftpd、httpd、lpd、snmpd、telnetd、smtpdなどのサーバデーモンや、同プロトコルのクライアント機能などを有している。

#### 【0046】

OCS126のプロセスは、オペレータ（ユーザ）と本体制御間の情報伝達手段となるオペレーションパネル（操作パネル）の制御を行う。OCS126は、

オペレーションパネルからのキー押下をキーイベントとして取得し、取得したキーに対応したキーイベント関数を S C S 1 2 2 に送信する O C S プロセスの部分と、アプリケーション 1 3 0 又はコントロールサービスからの要求によりオペレーションパネルに各種画面を描画出力する描画関数やその他オペレーションパネルに対する制御を行う関数などが予め登録された O C S ライブラリの部分とから構成される。この O C S ライブラリは、アプリケーション 1 3 0 およびコントロールサービスの各モジュールにリンクされて実装されている。なお、O C S 1 2 6 のすべてをプロセスとして動作させるように構成しても良く、あるいは O C S 1 2 6 のすべてを O C S ライブラリとして構成しても良い。

#### 【 0 0 4 7 】

アプリケーション 1 3 0 は、ページ記述言語 ( P D L ) 、 P C L およびポストスクリプト ( P S ) を有するプリンタ用のアプリケーションであるプリンタアプリ 1 1 1 と、コピー用アプリケーションであるコピーアプリ 1 1 2 と、ファクシミリ用アプリケーションであるファックスアプリ 1 1 3 と、スキャナ用アプリケーションであるスキャナアプリ 1 1 4 と、ネットワークファイル用アプリケーションであるネットファイルアプリ 1 1 5 と、工程検査用アプリケーションである工程検査アプリ 1 1 6 とを有している。これらの各アプリは、その起動時に V A S 1 4 0 に対して自プロセスのプロセス I D とともにアプリ登録要求メッセージを送信し、アプリ登録要求メッセージを受信した V A S 1 4 0 によって、起動したアプリに対する登録が行われるようになっている。

#### 【 0 0 4 8 】

アプリケーション 1 3 0 の各プロセス、コントロールサービスの各プロセスは、関数呼び出しとその戻り値送信およびメッセージの送受信によってプロセス間通信を行いながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを実現している。

#### 【 0 0 4 9 】

このように、実施の形態 1 にかかる複合機 1 0 0 には、複数のアプリケーション 1 3 0 および複数のコントロールサービスが存在し、いずれもプロセスとして動作している。そして、これらの各プロセス内部には、一又は複数のスレッドが

生成されて、スレッド単位の並列実行が行われる。そして、コントロールサービスがアプリケーション 130 に対し共通サービスを提供しており、このため、これらの多数のプロセスが並列動作、およびスレッドの並列動作を行って互いにプロセス間通信を行って協調動作をしながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを提供するようになっている。また、複合機 100 には、サードベンダなどの第三者がコントロールサービス層の上のアプリケーション層に新規アプリ 117、118 を開発して搭載することが可能となっている。図 1 では、この新規アプリ 117、118 を搭載した例を示している。

#### 【0050】

なお、実施の形態 1 にかかる複合機 100 では、複数のアプリケーション 130 のプロセスと複数のコントロールサービスのプロセスとが動作しているが、アプリケーション 130 とコントロールサービスのプロセスをそれぞれ単一の構成とすることも可能である。また、各アプリケーション 130 は、アプリケーションごとに追加又は削除することができる。更に、実施の形態 1 にかかる複合機 100 では、VAS 140 に加えてダイナミックリンクライブラリ (DLL) を採用することが可能である。

#### 【0051】

図 2 に複合機 100 のハードウェア構成例を示す。

#### 【0052】

複合機 100 は、コントローラ 160 と、オペレーションパネル 175 と、ファックスコントロールユニット (FCU) 176 と、プリンタ等の画像形成処理に特有のハードウェア資源であるエンジン部 177 とを含む。コントローラ 160 は、CPU 161 と、システムメモリ 162 と、ノースブリッジ (NB) 163 と、サウスブリッジ (SB) 164 と、ASIC 166 と、ローカルメモリ 167 と、HDD 168 と、ネットワークインターフェースカード (NIC) 169 と、SD カード用スロット 170 と、USB デバイス 171 と、IEEE 1394 デバイス 172 と、セントロニクス 173 とを含む。なお、メモリ 162、167 は RAM、ROM 等を含む。FCU 176 およびエンジン部 177 は、コ

ントローラ 160 の A S I C 166 に P C I バス 178 で接続されている。

#### 【0053】

C P U 161 が、複合機 100 にインストールされるアプリケーション、コントロールサービス等のプログラムを、メモリから読み出して実行する。

#### 【0054】

図 3 は、実施の形態 1 にかかる複合機 100 の V A S 140 の構成と、V A S 140 と各アプリ、コントロールサービス層 150 および汎用 O S 121 との関係を示すブロック図である。なお、図 3 では、アプリケーション 130 の例として、プリンタアプリ 111、コピーアプリ 112、新規アプリ 117、118 を示しているが、更にアプリを追加することもできる。

#### 【0055】

仮想アプリケーションサービス (V A S) 140 のプロセスには、デイスパッチャ 144 と、制御スレッド 143 と、ラッピング処理スレッド 141 と、バージョン管理スレッド 142 とが動作している。

#### 【0056】

デイスパッチャ 144 は、アプリケーション 130 やコントロールサービスからのメッセージ受信を監視し、受信したメッセージに応じて制御スレッド 143、ラッピング処理スレッド 141、バージョン管理スレッド 142 に処理要求を行うものである。実施の形態 1 の複合機 100 では、デイスパッチャ 144 は、各アプリが起動したときにアプリからのアプリ登録要求メッセージを受信すると、そのアプリ登録要求メッセージを制御スレッド 143 に送信する。

#### 【0057】

制御スレッド 143 は、デイスパッチャ 144 からアプリ登録要求メッセージを受信してアプリ登録処理を行う。ここで、アプリ登録処理とは、R A M 210 にアプリ登録テーブル (図示せず) を生成し、アプリ登録要求メッセージを送信したアプリの識別情報であるアプリ I D をアプリ登録テーブルに記録する処理をいう。また、制御スレッド 143 は、H D 200 に格納されたラッピング処理情報ファイル 201 を参照して、アプリ登録要求を行ったアプリについて、ラッピング処理情報が記録されているか否かをチェックする。

**【0058】**

ラッピング処理スレッド141の機能概要は次のとおりである。

**【0059】**

コントロールサービスが提供する関数（API）がバージョンアップして引数の追加等がされたが、アプリ側でそのコントロールサービスに対して使用するその関数はバージョンアップ前のものであったとしても、ラッピング処理スレッド141がアプリ側から呼び出された関数の変更を行うことにより、バージョン差を吸収するというものである。

**【0060】**

更に、ラッピング処理スレッド141は、ラッピング処理情報ファイル201を参照することにより、コントロールサービス層150からアプリに対して通知されるメッセージを選択する機能を有している。特に、複合機100のシステムに大きく影響を与えるインタフェースについては、サードベンダなどの第三者に対し隠蔽しておき、直接コントロールサービスに対してアクセスすることを回避する。これにより、複合機のセキュリティ面および障害発生の未然防止を行うことができる。

**【0061】**

また、バージョン管理スレッド142は、制御スレッド143からの処理要求を受け、アプリがVAS140に対して使用する関数のバージョンと、実際のVAS140が有する関数のバージョンとの違いを検出し、バージョン管理テーブル211を参照することにより、発生したバージョン差がVAS140によってサポート可能な範囲か否かを判定する。

**【0062】**

図4は、HD200に格納されるラッピング処理情報ファイル201の内容例を示す説明図である。図4に示すように、ラッピング処理情報ファイル201は、コントロールサービス層150からアプリに対するメッセージ（イベント）の非通知設定を行うための情報ファイルであり、複合機100のシステムに大きく影響を与えるインタフェースについては第三者に対して隠蔽するよう、メッセージごとに任意の非通知設定を行うことができる。図4のイベントAとCは、非通

知設定がなされているのでアプリには通知されず、また、イベントBのように非通知設定がなされていないイベントについては、通常通り通知が行われる。

#### 【0063】

図5は、バージョン管理テーブル211の内容例を示す説明図である。なお、このテーブルは、VAS140の実行ファイルに含める形で実装でき、VAS140が実行される際には例えばRAM210に展開される。実行ファイルにテーブル211を含めるには、例えば、図5に示すバージョン情報をインクルードファイルとして作成しておき、作成したプログラムにインクルードしてコンパイルする。また、このテーブルを別ファイルとし、これを複合機100に格納しておき、VAS140が適宜このファイルを参照してもよい。

#### 【0064】

図5に示すように、このバージョン管理テーブル211には、VAS140の全体バージョン番号と、VAS140がサポート可能なアプリの全体バージョンの範囲が記録されている。なお、サポート可能な範囲は、例えば、複合機100で使用されるVAS140のバージョンより一定範囲前のバージョンまでである。

#### 【0065】

なお、ここで、VAS140の全体バージョンとは、VAS140が提供する関数のセットのバージョンであり、アプリの全体バージョンとは、アプリがVAS140に対して使用する関数のセットのバージョンである。なお、アプリがVAS140に対して使用する関数のセットのバージョン、アプリを作成するときに使用したVAS140の関数セットのバージョンと同じである。

#### 【0066】

更に、バージョン管理テーブル211は、関数毎に、VAS140の関数のバージョン番号とその関数がサポートできる、アプリ側がVAS140に対して使用する関数のバージョンの範囲を含んでいる。なお、VAS140は上記のように各機能をスレッドとして実装してもよいし、1つのプロセスとして動作するようにしてもよい。

#### 【0067】

次に、このように構成された複合機100のVAS140によるラッピング処理とバージョン管理について説明する。まず、ラッピング処理について説明する。

#### 【0068】

図6は、VAS140のラッピング処理スレッドによるラッピング処理の手順を示すフローチャートである。

#### 【0069】

コントロールサービスが提供する関数がバージョンアップされた場合、その関数と、アプリがそのコントロールサービスに対して使用する関数との間にバージョン差が生じる。このような場合に、上記関数間の違いを吸収するためのVAS140の処理を図6を用いて説明する。

#### 【0070】

なお、VAS140は、各コントロールサービスに対する関数毎に、その前のバージョンに対して、関数がバージョンアップされているか否か、バージョンアップされているとすればそのバージョンアップがどのようなものかを示す情報を有している。この情報は、VAS140のプログラムの作成時に、例えば図7に示すテーブルの形式でプログラムに含めることができる。なお、バージョンアップの態様としては、例えば、関数の追加、分割、削除、変更、統合があり、また、引数の追加、分割、削除がある。

#### 【0071】

まず、ステップS601において、アプリからの処理要求に対応する関数がバージョンアップされているか否かを図7の情報を参照することによりチェックする。バージョンアップされていなければ通常の処理を行う（ステップS604）。バージョンアップされていれば、そのバージョンアップの態様が、関数の分割又は引数の追加かどうかを判断する（ステップS602）。

#### 【0072】

ここで、関数の分割又は引数の追加、のいずれでもない場合には、バージョンアップが関数の削除、変更、統合等である。この場合には、アプリからの要求に対応する関数がコントロールサービス側に存在しなくなるため、アプリからの関



数呼び出しがあってもその処理を行うことができない。このため、アプリに対してNGであることを通知し（ステップS605）、終了を処理する（ステップS606）。

#### 【0073】

一方、ステップS602において、関数の分割、引数の追加である場合、対応する関数の数、引数の数が増えるため、増えた部分にダミー関数、又は、ダミー引数を補填する（ステップS603）、そして、通常の処理を行う（ステップ604）。なお、ダミー引数を補充する場合のVAS140の記述例を図8に示す。図8に示すように、アプリ側から要求を受ける関数“API\_for\_Apli\_Nol(int arg1, int arg2)”を、引数dummyを含む“API\_for\_XCS\_Nol(arg1, arg2, dummy)”に変換している。

#### 【0074】

なお、関数が追加されたことによりコントロールサービスがバージョンアップしている場合には、既存の関数自体は変化しないことから、追加された機能はアプリから使えないが、アプリからの関数呼び出しには影響を与えない。

#### 【0075】

次に、コントロールサービスの特定のインタフェースを隠蔽するための処理について説明する。コントロールサービスからアプリに対して全てメッセージ通知（例えば、イベント通知）が行われるとすると、複合機100のシステムに大きく影響を与えるコントロールサービスに対して直接アクセスすることが可能となり、複合機のセキュリティ面および障害発生を未然に防止する面から好ましくない。

#### 【0076】

そこで、実施の形態1では、非通知とするイベントを、図4のラッピング処理情報ファイル201に予め設定する。この設定はどのようにして行ってもよい。例えば、複合機100外でラッピング処理情報ファイル201を作成し、複合機100に格納してもよいし、VAS140が設定のための画面をオペレーションパネルに表示し、そこから設定してもよい。また、VAS140の実行ファイルに含めるようにして実装してもよい。

**【0077】**

図9は、VAS140によりラッピング処理を行う手順を示すフローチャートである。

**【0078】**

あるコントロールサービスからアプリに向けてメッセージ通知がなされる場合、ラッピング処理スレッド141がラッピング処理情報ファイル201を参照する（ステップS701）。そのメッセージに対して非通知設定がなされているかどうかをチェックし（ステップS702）、そのメッセージに対して非通知設定がなされていれば、ラッピング処理スレッド141はそのメッセージをアプリに通知しない（ステップS703）。また、上記ステップS701において、上記のメッセージに対してラッピング処理情報ファイル201内の設定自体が行われていないか、非通知設定されていない場合は、通常通りアプリにメッセージが通知される（ステップS704）。

**【0079】**

ここでは、特定のイベントをアプリに通知するか否かを任意かつ事前に設定できるようにしたため、複合機のセキュリティ面および障害発生を未然に防止することが可能になる。また、実施の形態1では、新規アプリケーションを開発するサードベンダなどの第三者に特定のインタフェースを開示する必要があるため、開示できるインタフェースを自由に選択できるようにすることは非常に重要である。

**【0080】**

次に、アプリがVAS140に対して使用する関数（API）と、VAS140が有する関数とのバージョン差をチェックする手順について説明する。上記のように、VAS140により、アプリとコントロールサービス間での関数のバージョン差を吸収するが、VAS140自体のバージョンアップにより、VAS140がアプリに対して提供する関数と、アプリがVAS140に対して使用する関数との間でバージョン差が生じることがある。これから説明する処理は、バージョン差をチェックし、VAS140がサポートできる範囲のバージョン差であればVAS140の通常の処理を行うというものである。

**【0081】**

この処理では、アプリ自身が、全体バージョン情報と、VAS140に対して使用する関数毎のバージョン情報を保持し、アプリが起動したときに、その情報をプロセス間通信によりVAS140に通知している。アプリ側のこのバージョン情報は、例えばアプリの作成時にアプリに含められる。

**【0082】**

VAS140によるバージョンチェック時期としては、アプリが実行される前であれば良く、例えばここではアプリ登録時に行ったが、これに限定されない。例えば、アプリを仮起動させて、そのときにVAS140がバージョンチェックを行ってもよい。仮起動とは、VAS140が、アプリとのプロセス間通信によりアプリの情報を取得するためだけにアプリを起動することである。なお、この場合のアプリは仮起動可能なように作られている。

**【0083】**

登録時には次のような処理が行われる。

**【0084】**

デイスパッチャ144が、起動されたアプリからアプリ登録要求メッセージを受信すると、アプリ登録要求メッセージをそのアプリのプロセスIDとともに制御スレッド143に受け渡す。制御スレッド143は、アプリ登録要求メッセージとプロセスIDをデイスパッチャ144から受信すると、アプリを識別するアプリIDを決定して、RAM210などに格納したアプリ登録テーブル（図示せず）にアプリIDを記録することにより、アプリ登録が行われる。なお、アプリIDは、コピーアプリ112、プリンタアプリ111など既存のアプリケーションについては、予め定められており、各アプリIDをVAS140の内部で保持している。また、サードベンダなどが開発した新規アプリ117、118については、最初の起動時におけるアプリ登録処理の中で決定される。

**【0085】**

以下、全体バージョンのチェックを行う場合と、関数毎のバージョンチェックを行う場合を分けて説明するが、全体バージョンチェックにおいてサポート範囲外と判断された場合にのみ、関数毎のバージョンチェックを行うようにしてもよ

い。

#### 【0086】

まず、全体バージョンのチェックを行う場合について図10を参照して説明する。

#### 【0087】

アプリ登録の要求があった場合（ステップS801）、まず、VAS140はアプリが有している全体バージョン番号を取得する（ステップS802）。

#### 【0088】

VAS140は、バージョン管理テーブル211の全体バージョンのサポート範囲とアプリの全体バージョン番号とを比較し、サポート範囲内か否かを判断する（ステップS803）。例えば、現在使用されているVAS140がサポートできるバージョンの範囲が1.0～1.6であれば、この範囲内のバージョン番号であるときにサポート範囲内と判定する。この場合、当該アプリに対して「OK」が通知され（ステップS804）、正常にアプリ登録され（ステップS805）、通常の複合機100における処理が行われる（ステップS806）。また、ステップS803でアプリから取得したバージョン番号がサポート範囲外である場合には、当該アプリに対して「NG」が通知され（ステップS807）、アプリ登録が終了する（ステップS808）。

#### 【0089】

次に、バージョンを関数毎にチェックする場合について説明する。図11にその処理手順を示す。

#### 【0090】

図11の場合も、図10と同様にVAS140によるバージョンチェック時期として、アプリが実行される前であれば良く、ここではアプリ登録時を想定している。

#### 【0091】

アプリ登録の要求があった場合（ステップS901）、VAS140はアプリから、アプリがVAS140に対して使用している関数毎のバージョン情報を取得する（ステップS902）。そして、VAS140は、バージョン管理テーブ

ル 211 を参照し、アプリが VAS 140 に対して使用している関数毎に、アプリが VAS 140 に対して使用する関数のバージョン番号と、サポート範囲とを比較することにより、当該関数が VAS 140 のサポート範囲内か否かを判断する（ステップ S903）。例えば、アプリが VAS 140 に対して使用している関数番号が「2」、「3」であり、そのバージョン番号がそれぞれ「バージョン 1.1」と「バージョン 2.0」である場合には、各関数がサポート範囲内となる。

#### 【0092】

図 11 の処理において、ステップ S903 においてアプリケーションが VAS 140 に対して使用している全ての関数についてサポート範囲内と判定された場合は、上記した図 10 の A 以降と同様に処理され、ステップ S903 においてサポート範囲内でないと判定された場合は、図 10 の B 以降と同様に処理される。なお、NG の場合に、VAS 140 は、どの関数が NG であるかをオペレーションパネルに表示してもよい。

#### 【0093】

アプリが使用していない関数のバージョンが VAS 140 側で変更された場合でも、VAS 140 のバージョンは変更されるため、全体バージョンの比較をただだけではバージョン差によってアプリの実行が不可となる場合があり得る。そこで、図 11 に示した処理のように、アプリが VAS 140 に対して使用している関数のバージョンを関数毎にチェックすることにより、上記の場合でもアプリの実行に問題がないことを判断できる。また、関数毎のチェックでサポート範囲外と判定された場合、その原因となった関数を容易に特定できる。

#### 【0094】

このように、実施の形態 1 にかかる複合機 100 では、VAS 140 のラッピング処理スレッド 141 が、バージョン差を吸収するので、コントロールサービスの関数がバージョンアップしたとしても、アプリをコンパイルし直す必要がなくなる。また、VAS 140 は、コントロールサービスからアプリに対するメッセージを取捨選択できるため、複合機 100 のシステムに大きく影響を与えるようなインタフェースについては、第三者に対して隠蔽することができる。

**【0095】**

また、実施の形態1にかかる複合機100では、VAS140が、アプリから取得したバージョン情報を用いてバージョンチェックをできるので、安全にアプリを実行できる。また、実施の形態1にかかる複合機100では、上記したようにVAS140に加えてダイナミックリンクライブラリ(DLL)を採用することも可能であり、その場合はアプリケーションとコントロールサービス間のバージョン差の吸収力が増大するとともに、インタフェースを隠蔽することができるためインタフェースの秘匿性をより強固に確保することが可能となる。

**【0096】**

(実施の形態2)

実施の形態1にかかる複合機100は、VAS140が全アプリケーションに対して1つだけ存在するものであった。この実施の形態2にかかる複合機では、各アプリごとに一つのVASが起動し、各VASは対応するアプリに対してのみバージョン管理およびラッピング処理を行うものである。

**【0097】**

図12は、実施の形態2にかかる複合機800の構成を示すブロック図である。図12に示すように、複合機800では、複数の仮想アプリケーションサービス(VAS)841～848がアプリケーション130の各アプリごとに動作している点が、実施の形態1にかかる複合機100と異なっている。

**【0098】**

VAS841～848は、プリンタアプリ111、コピーアプリ112、ファックスアプリ113、スキャナアプリ114、ネットファイルアプリ115、工程検査アプリ116、新規アプリ117および118に対応して、バージョン管理およびラッピング処理を行うようになっている。

**【0099】**

図13は、実施の形態2にかかる複合機800のVAS841～848の構成と、VAS841～848と各アプリ、コントロールサービス層150および汎用OS121との関係を示すブロック図である。なお、図12では、アプリケーション130として、プリンタアプリ111、コピーアプリ112、新規アプリ

117、118の例を示し、更にこれら各アプリに対応したVAS 841、842、847および848を例として示したが、他のアプリの場合も同様の構成となる。

#### 【0100】

また、実施の形態2にかかる複合機800では、実施の形態1の複合機100と異なり、図13に示すように、各VAS 841～848と各アプリとの間にはVAS制御プロセス（デーモン）801が動作している。

#### 【0101】

このVAS制御プロセス（デーモン）801は、各アプリからアプリ登録要求メッセージを受信してアプリ登録処理を行うとともに、アプリ登録要求を行ったアプリに対応したVAS 841～848を生成する。また、各スレッドに対して処理の指示を行う。

#### 【0102】

そして、仮想アプリケーションサービス（VAS）841～848のプロセスには、デイスパッチャ144と、ラッピング処理スレッド141と、バージョン管理スレッド142とが動作している。

#### 【0103】

デイスパッチャ144は、アプリケーション130やコントロールサービスからのメッセージ受信を監視し、例えば、受信したメッセージに応じてラッピング処理スレッド141、あるいはバージョン管理スレッド142に対して処理要求を行うものである。実施の形態2の複合機800では、デイスパッチャ144は、VAS制御プロセス801から、アプリID、アプリのプロセスIDとともに、バージョン管理要求メッセージ又はラッピング処理要求メッセージを受信するようになっている。デイスパッチャ144は、バージョン管理要求メッセージを受信したときには、アプリID、アプリのプロセスIDとともに受信したバージョン管理要求メッセージをバージョン管理スレッド142に送信し、ラッピング処理要求メッセージを受信したときには、アプリID、アプリのプロセスIDとともに受信したラッピング処理要求メッセージをラッピング処理スレッド141に送信する。バージョン管理スレッド142とラッピング処理スレッド141の

処理内容は第 1 の実施の形態と同様である。

#### 【0104】

このように実施の形態 2 にかかる複合機 800 によれば、第 1 の実施の形態と同様の効果がある。更に、実施の形態 2 にかかる複合機 800 では、VAS 841～848 は起動されるアプリケーション 130 ごとに別個に起動されるので、複数のアプリケーション 130 に対する判断処理を各アプリケーション 130 に対応する VAS 841～848 で並列に実行することができ、処理が効率的に行えると共に、上記したバージョン管理やラッピング処理がアプリごとに行えるので、新規アプリの追加や交換時にも効率良く処理を行うことができる。

#### 【0105】

また、VAS の構成として上記の構成の他、図 14 (a)～(c) に示す構成をとることもできる。

#### 【0106】

図 14 (a) は、各アプリケーションに対して起動される VAS を、親 VAS の子プロセスとする場合であり、親 VAS 自体は画面制御権（ユーザインターフェース）を持たない。図 14 (b) は、親 VAS 自体が画面制御権（ユーザインターフェース）を持つ場合である。図 14 (c) は、各アプリケーションに対応する VAS の機能をスレッドとして起動する場合を示している。

#### 【0107】

なお、本発明は、上記の実施の形態に限定されることなく、特許請求の範囲内において、種々変更・応用が可能である。

#### 【0108】

##### 【発明の効果】

上記のように、本発明によれば、アプリケーションが使用している関数とコントロールサービス側の関数とで差異が生じた場合でも、アプリケーションのソースプログラムを書き換えることなく、関数呼び出しを行うことが可能となる。また、コントロールサービスからアプリに通知されるメッセージのうち特定のメッセージを秘匿することが可能となる。

##### 【図面の簡単な説明】



**【図 1】**

実施の形態 1 にかかる複合機の構成を示すブロック図である。

**【図 2】**

実施の形態 1 にかかる複合機のハードウェア構成を示すブロック図である。

**【図 3】**

実施の形態 1 にかかる複合機の V A S の構成と、V A S と各アプリ、コントロールサービス層および汎用 O S との関係を示すブロック図である。

**【図 4】**

H D に格納されるラッピング処理情報ファイルの内容例を示す説明図である。

**【図 5】**

バージョン管理テーブルの内容例を示す説明図である。

**【図 6】**

V A S によりラッピング処理を行う手順を示すフローチャートである。

**【図 7】**

バージョンアップの有無及びバージョンアップの態様を記録したテーブルを示す図である。

**【図 8】**

ダミー引数を補充する場合の V A S 1 4 0 の記述例を示す図である。

**【図 9】**

V A S により特定のメッセージを隠蔽するためのラッピング処理を行う手順を示すフローチャートである。

**【図 1 0】**

V A S により全体バージョンをチェックする手順を示すフローチャートである。

**【図 1 1】**

V A S により関数単位にバージョンをチェックする手順を示すフローチャートである。

**【図 1 2】**

実施の形態 2 にかかる複合機の構成を示すブロック図である。

## 【図 13】

実施の形態 2 にかかる複合機の VAS の構成と、VAS と各アプリ、コントロールサービス層および汎用 OS との関係を示すブロック図である。

## 【図 14】

VAS の構成例を示す図である。

## 【符号の説明】

- 100 複合機
- 101 白黒レーザプリンタ
- 102 カラーレーザプリンタ
- 103 ハードウェアリソース
- 110 ソフトウェア群
- 111 プリンタアプリ
- 112 コピーアプリ
- 113 ファックスアプリ
- 114 スキャナアプリ
- 115 ネットファイルアプリ
- 116 工程検査アプリ
- 117、118 新規アプリ
- 120 プラットホーム
- 121 汎用 OS
- 122 SCS
- 123 SRM
- 124 ECS
- 125 MCS
- 126 OCS
- 127 FCS
- 128 NCS
- 130 アプリケーション
- 140、841～848 仮想アプリケーションサービス (VAS)

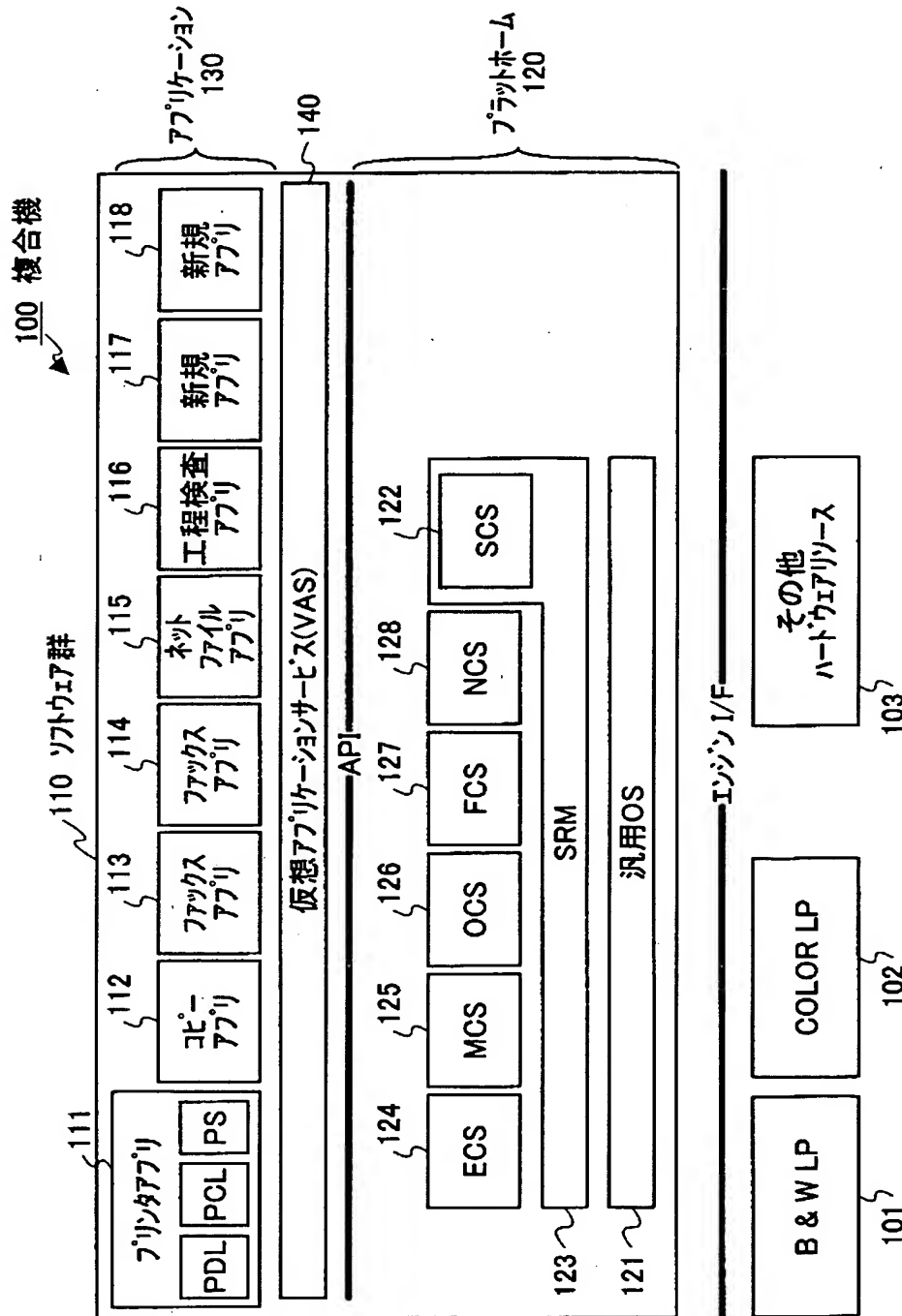
- 141 ラッピング処理スレッド
- 142 バージョン管理スレッド
- 143 制御スレッド
- 144 デイスパッチャ
- 150 コントロールサービス層
- 200 ハードディスク (HD)
- 201 ラッピング処理情報ファイル
- 210 RAM
- 211 バージョン管理テーブル
- 800 複合機
- 801 VAS制御プロセス

【書類名】

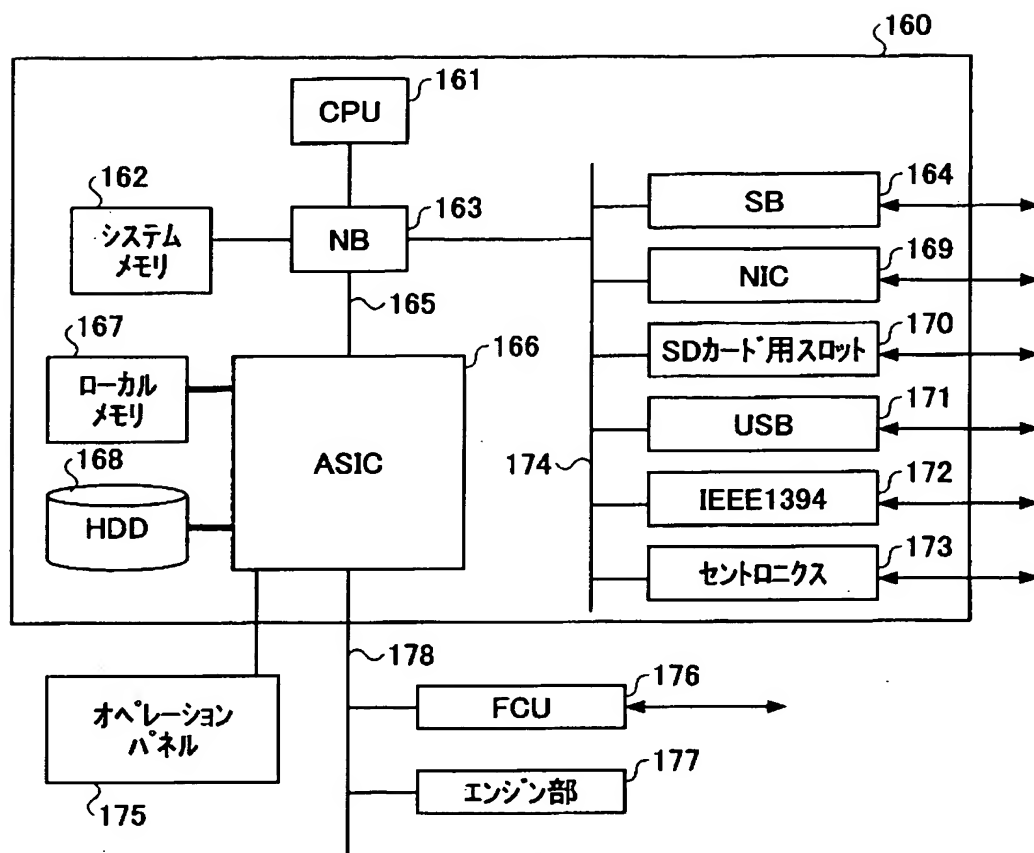
図面

【図 1】

実施の形態1にかかる複合機の構成を示すブロック図

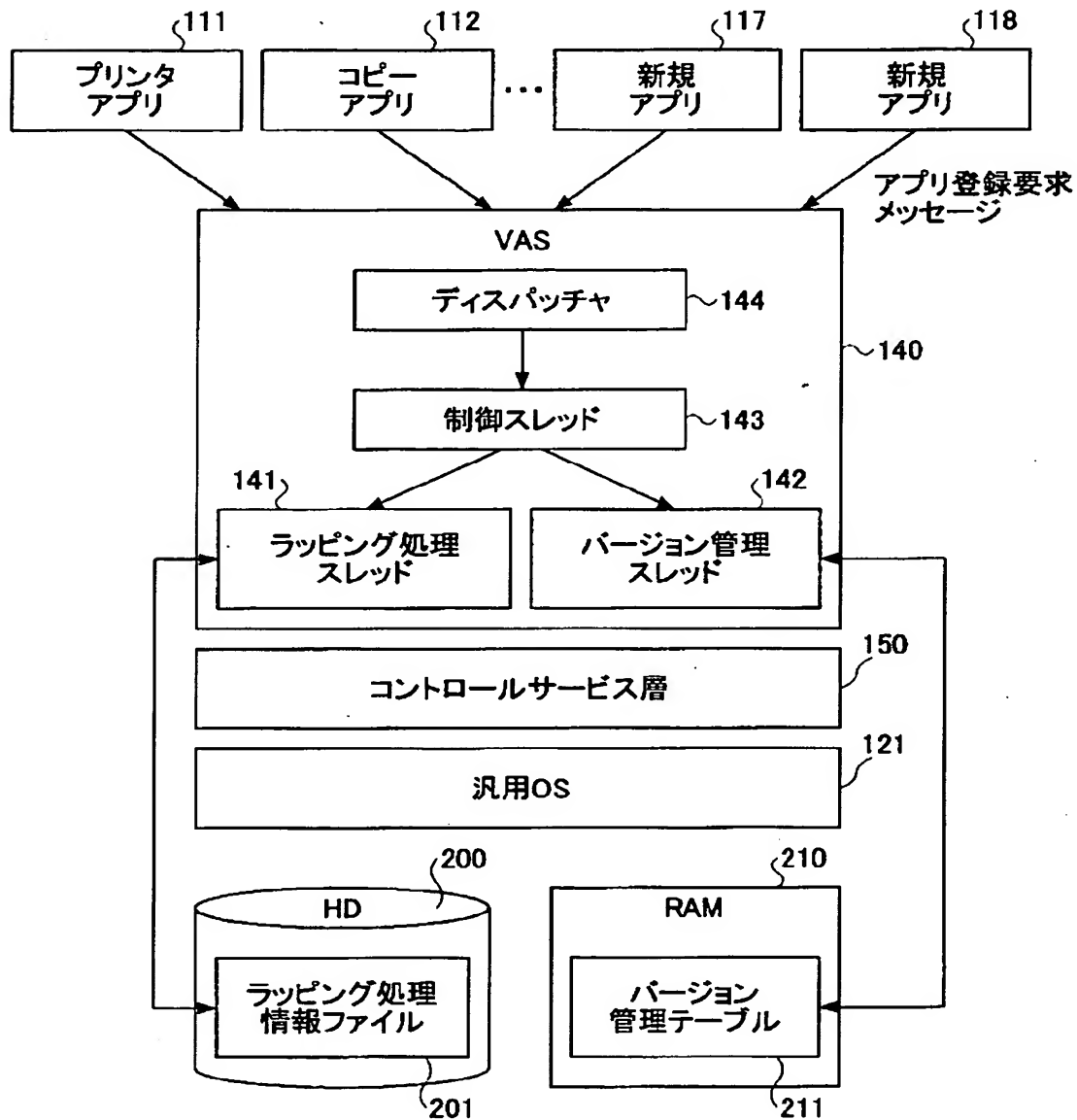


【図 2】

実施の形態1にかかる複合機の  
ハードウェア構成を示すブロック図

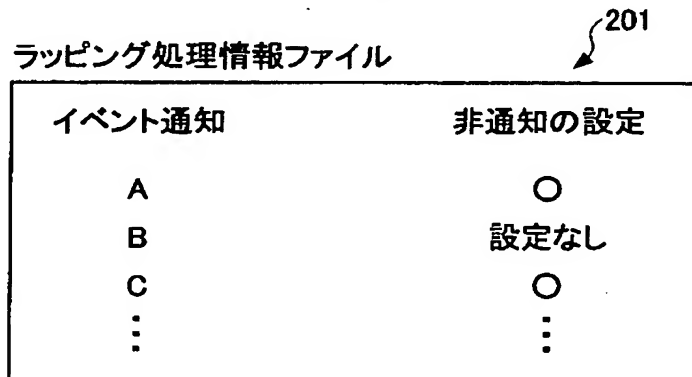
【図 3】

実施の形態1にかかる複合機のVASの構成と、VASと各アプリ、コントロールサービス層および汎用OSとの関係を示すブロック図



【図 4】

HDに格納されるラッピング処理情報ファイルの内容例を示す説明図



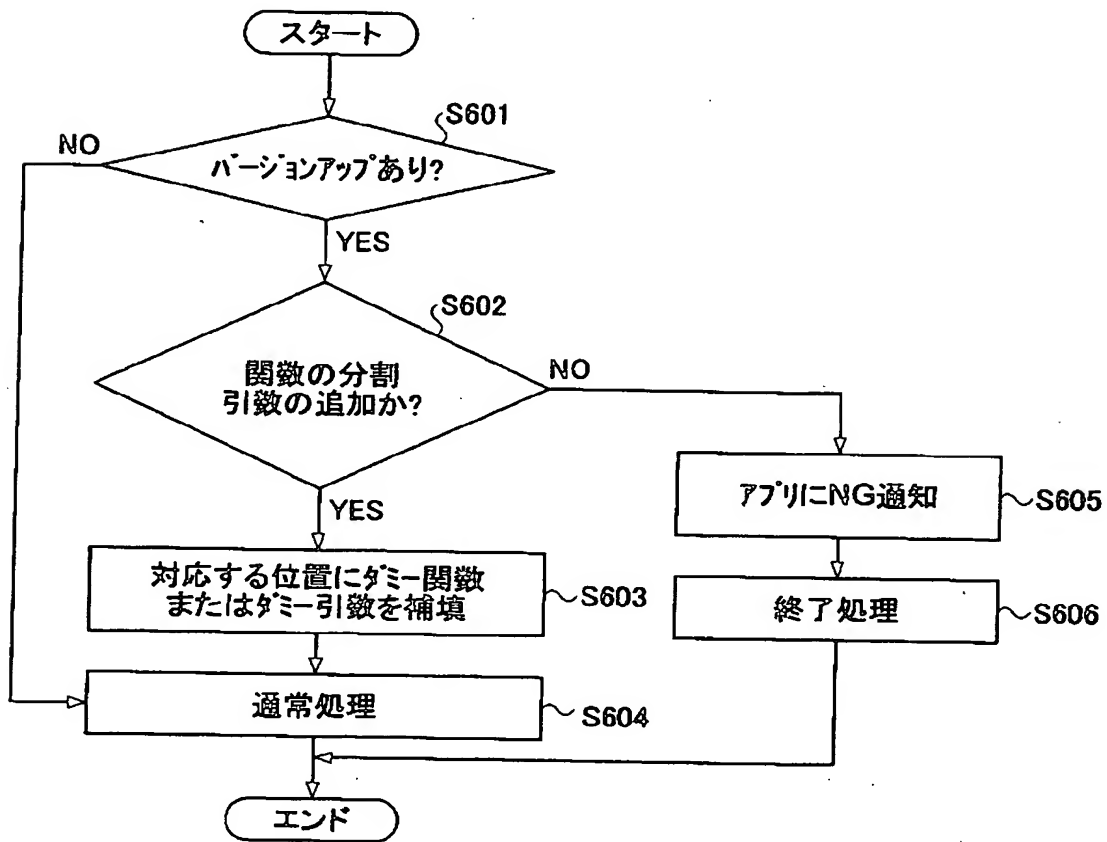
【図 5】

バージョン管理テーブルの内容例を示す説明図

全体バージョン番号2.3		サポート範囲2.0～2.3	
関数番号	バージョン番号	サポート範囲	使用の有無
1	1.4	1.0～1.4	×
2	1.2	1.0～1.2	○
3	2.3	2.0～2.3	○
4	1.1	1.0～1.1	×
⋮	⋮	⋮	⋮
n			

【図 6】

VASによりラッピング処理を行う手順を示すフローチャート





【図 7】

バージョンアップの有無及び  
バージョンアップの態様を記録したテーブルを示す図

関数番号	バージョン変更有無	関数分割	引数追加	...
1	1	0	1	
2	0	0	0	
3	0	0	0	
4	1	1	0	
⋮	⋮	⋮	⋮	

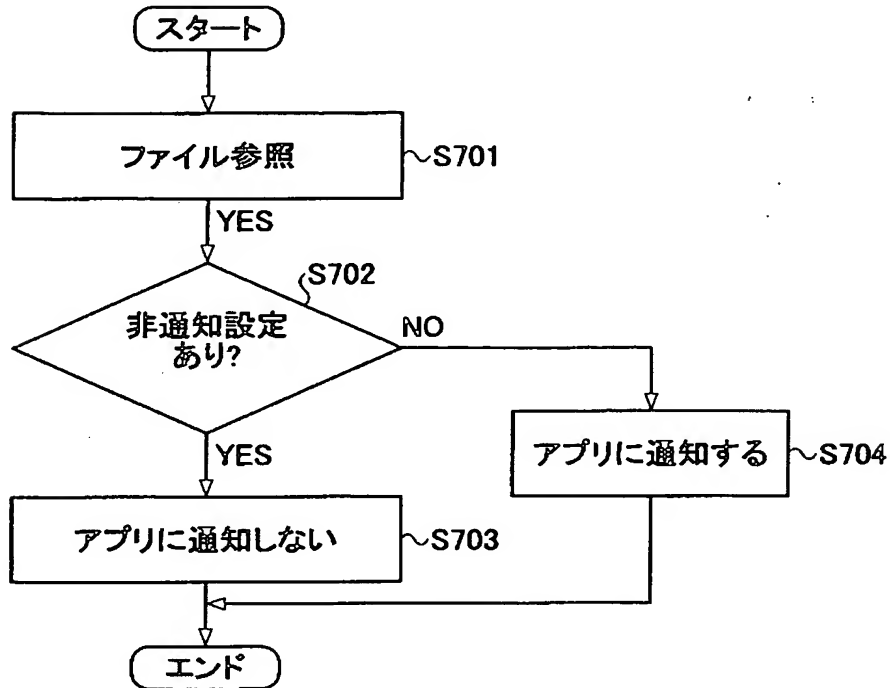
【図 8】

ダミー引数を補充する場合のVAS140の記述例を示す図

```
API_for_Apli_No1(int arg1, int arg2) {  
    int dummy = 0;  
    ...  
    API_for_XCS_No1(arg1, arg2, dummy);  
    ...  
}
```

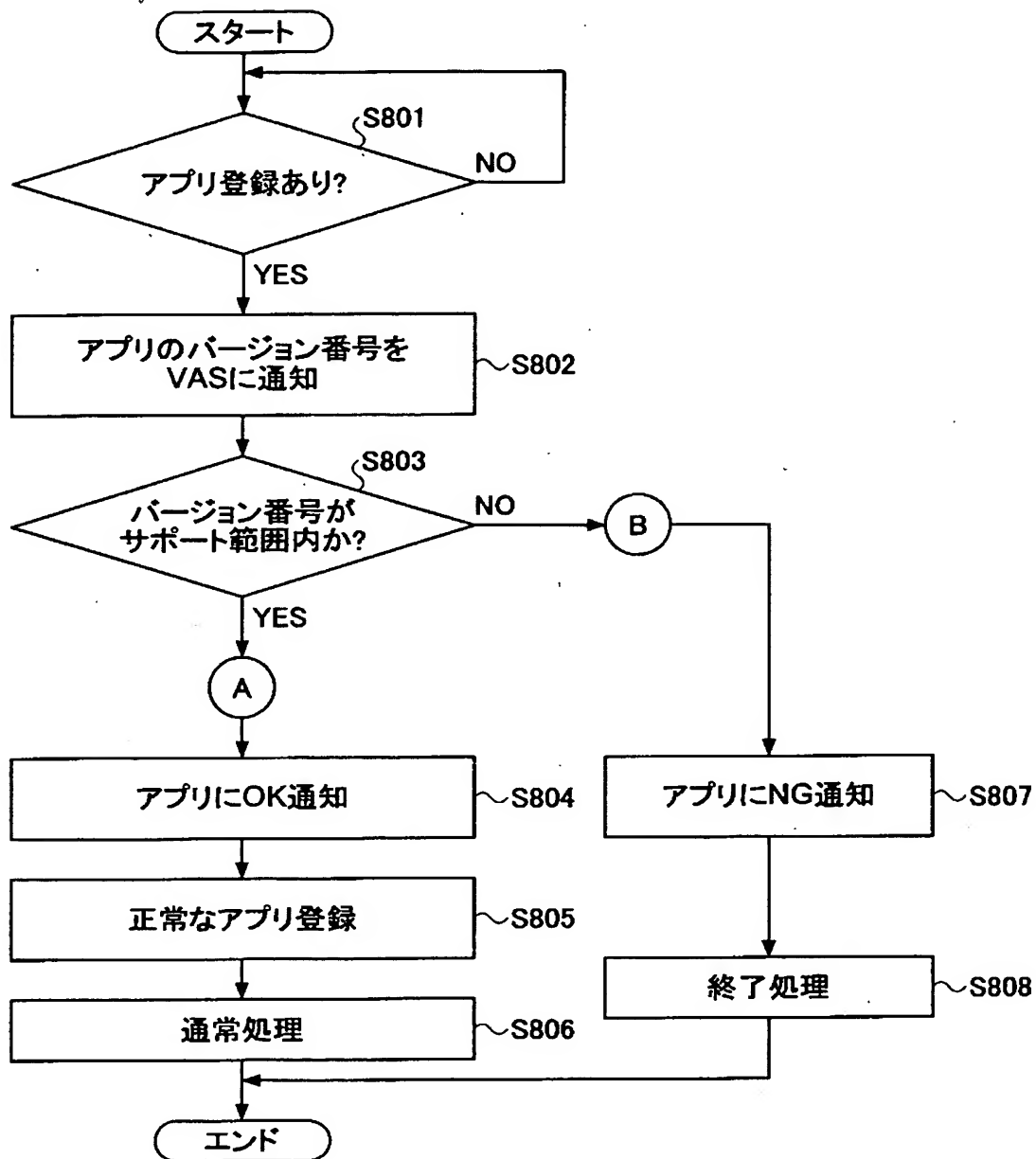
【図 9】

VASにより特定のメッセージを隠蔽するための  
ラッピング処理を行う手順を示すフローチャート



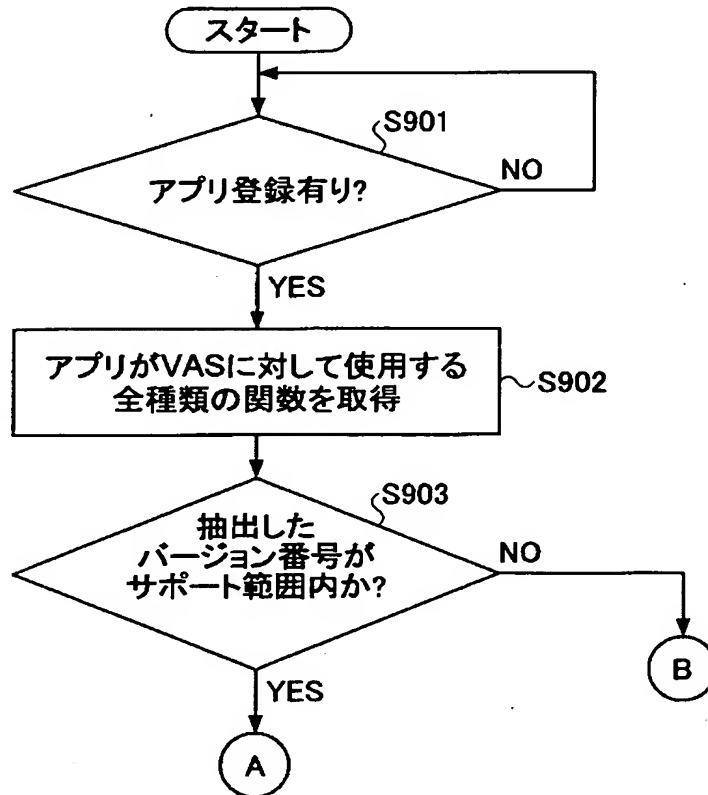
【図10】

VASにより全体バージョンをチェックする手順を示すフローチャート



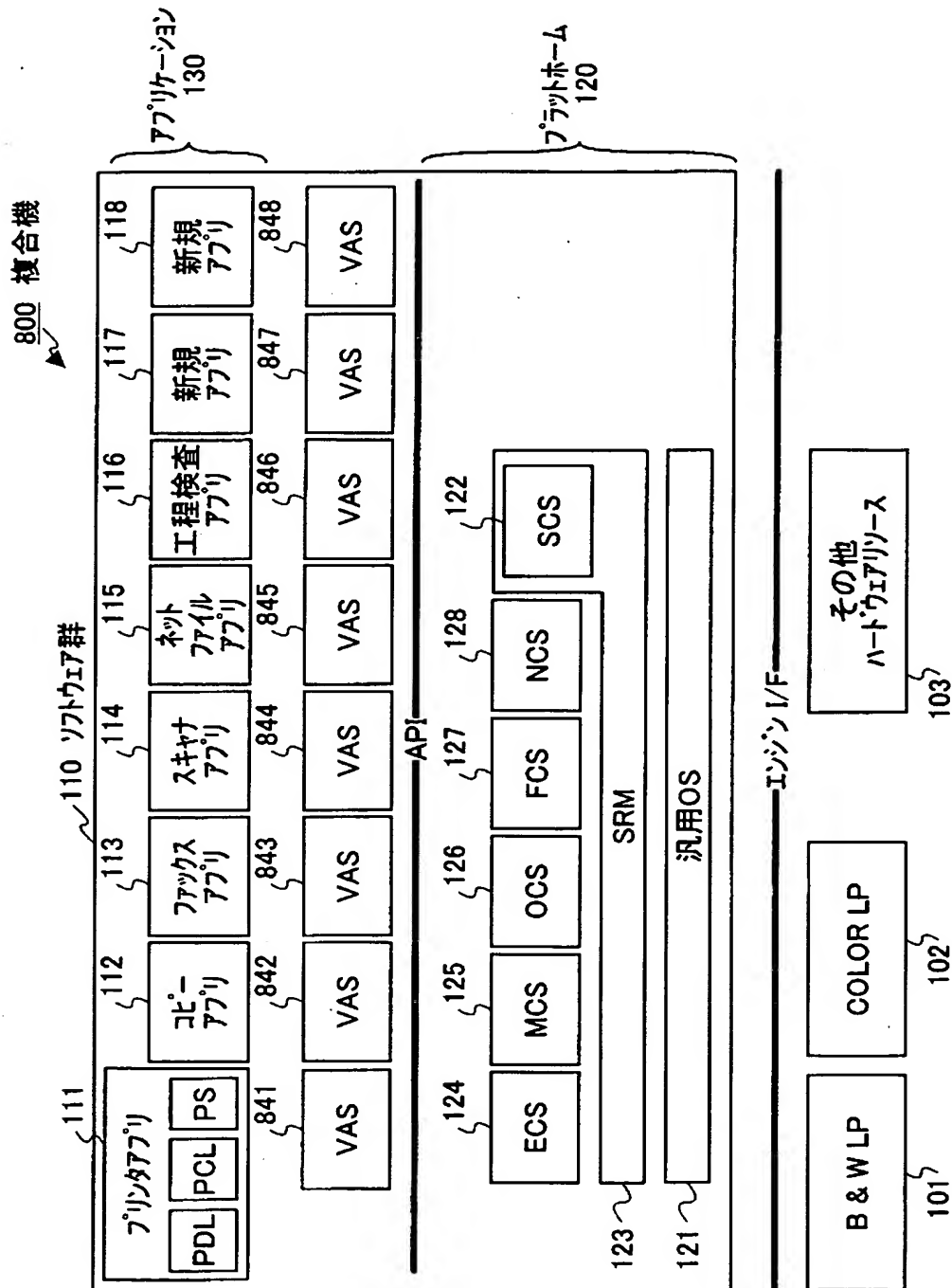
【図 11】

VASにより関数単位にバージョンをチェックする手順を示す  
フローチャート



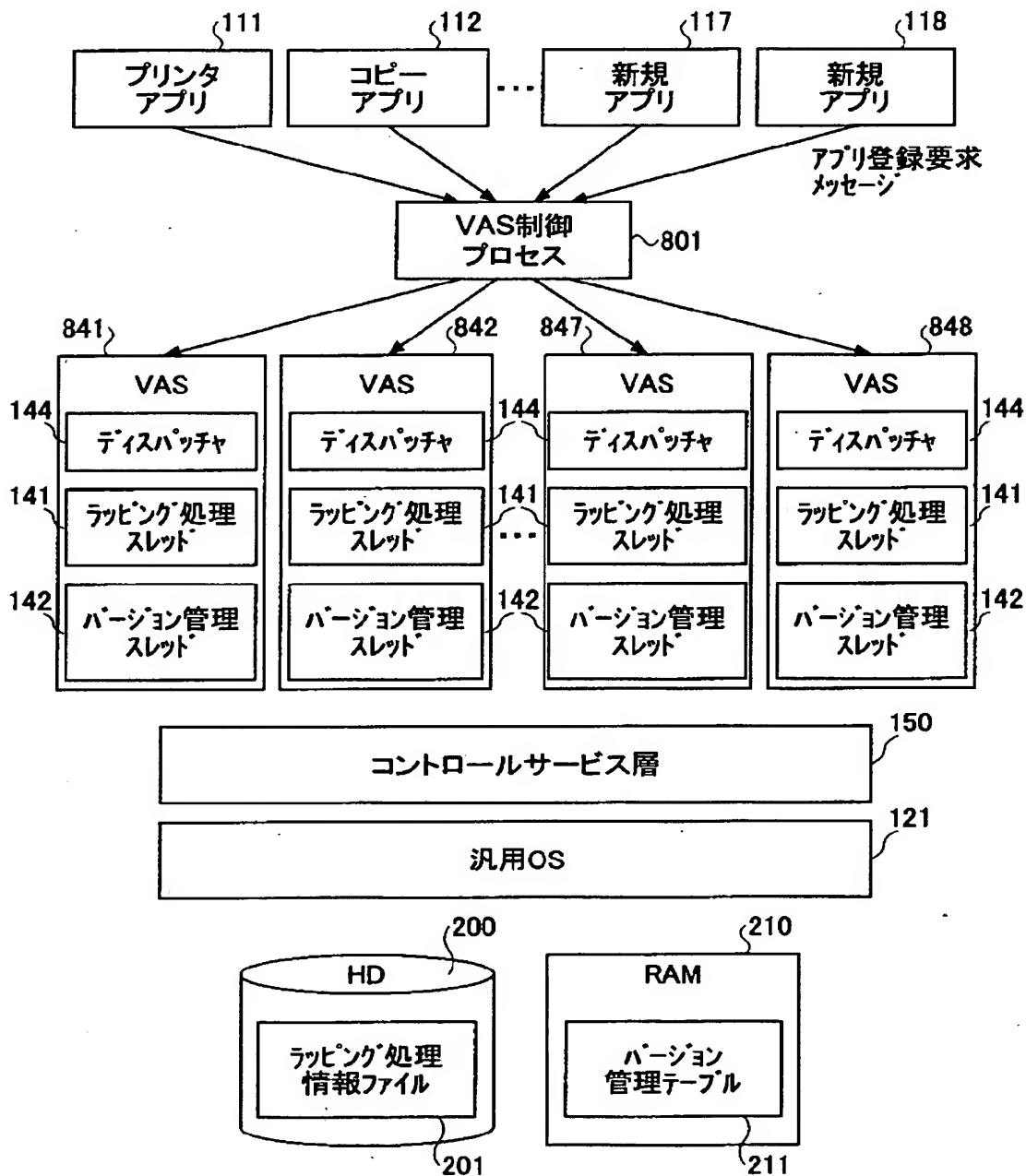
【図 12】

実施の形態2にかかる複合機の構成を示すブロック図



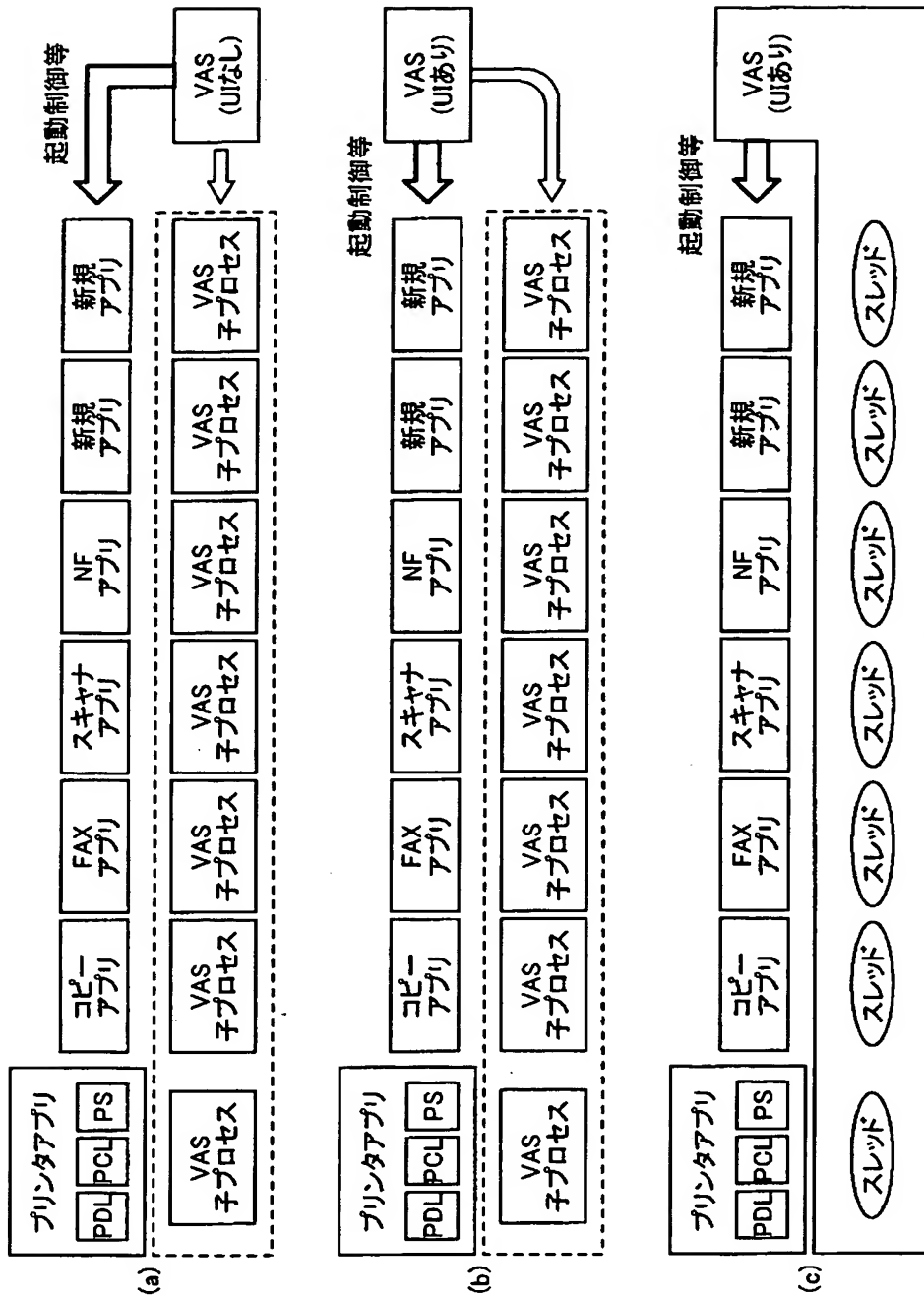
【図 13】

実施の形態2にかかる複合機のVASの構成と、VASと各アプリ、コントロールサービス層および汎用OSとの関係を示すブロック図



【図14】

VASの構成例を示す図



【書類名】 要約書

【要約】

【課題】 アプリケーションが使用している関数とコントロールサービス側の関数とで差異が生じた場合でも、関数呼び出しを行うことを可能とする技術を提供する。

【解決手段】 画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置において、アプリケーションから呼び出される関数を変換し、変換後の関数を用いてコントロールサービスへの関数呼び出しを行うラッピング処理手段を備える。

【選択図】 図 3



7  
特願 2 0 0 3 - 1 9 6 2 2 8

出 願 人 履 歴 情 報

識別番号

[ 0 0 0 0 0 6 7 4 7 ]

1. 変更年月日

1 9 9 0 年 8 月 2 4 日

[変更理由]

新規登録

住 所

東京都大田区中馬込 1 丁目 3 番 6 号

氏 名

株式会社リコー

2. 変更年月日

2 0 0 2 年 5 月 1 7 日

[変更理由]

住所変更

住 所

東京都大田区中馬込 1 丁目 3 番 6 号

氏 名

株式会社リコー